Configuration Manual

MSc Research Project
Data Analytics

Rajat
Student ID: x20232225

School of Computing
National College of Ireland

Supervisor: Dr. Catherine Mulwa

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Rajat		
Student ID:	X20232225		
Programme:	Msc Data Analytics	**Year:**	2022
Module:	Msc Research Project		
Supervisor:	Dr. Catherine Mulwa		
Submission Due Date:	15/08/2022		
Project Title:	Configuration Manual		
Word Count:	1157 **Page Count** 11		

Signature: Rajat

Date: 15/08/2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Rajat
X20232225

Introduction

This manual contains all the details on the software tools needed to carry out to “Identify the foliar diseases in apple trees using deep learning techniques” from scratch to end and also all the other details that could not be shared in the technical report are also provided in this configuration manual. The main objective of this research is What transfer learning techniques could be most effective for dealing with the disease-related issue? Which ones perform best and why? To answer this, multiple deep learning techniques have been used to identify diseases such as LittleVGG, MobileNet, and EffiecientNet. The dataset used for this study has been taken from the open-source platform Kaggle. This configuration manual, which comes with the project report, helps in properly understanding the project's configuration procedure.

1 Hardware and Software Specifications

To classify the apple foliar disease below are the hardware and software used to train the models on the image dataset.

1.1 Hardware Requirement

Table 1: System Specification

System	Specification
OS edition	macOS
Processor	Apple M1 chip
RAM	16 GB
CPU/GPU	8-core CPU, 8-core GPU
Neural Engine	16-core Neural Engine

1.2 Software Requirement

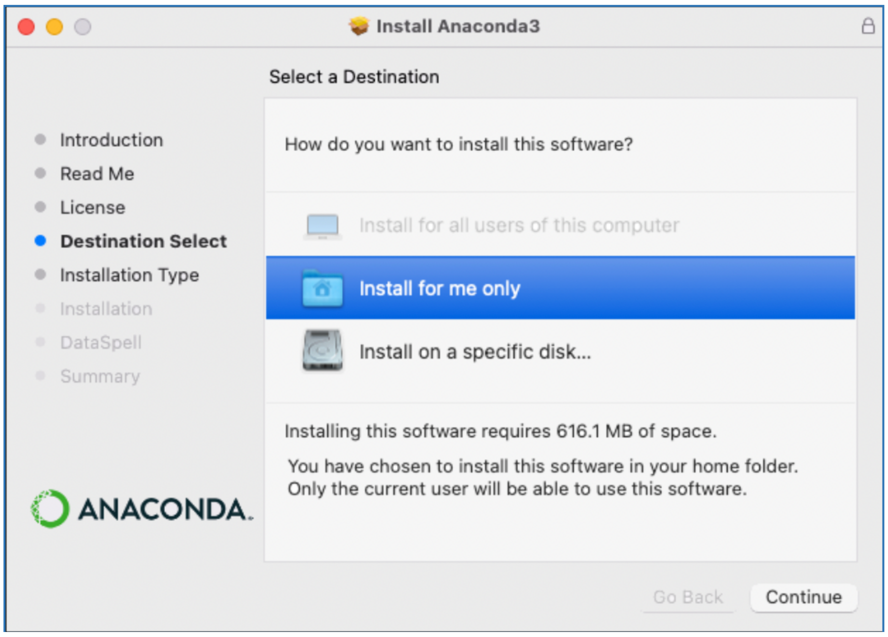
Table 2: Software Requirement

Programming Language	Python
Development Tool	Anaconda Navigator, Jupyter Notebook
Other Tools	Microsoft Excel and PowerPoint

2 Software Installation Guide

2.1 Installation steps of Anaconda Navigator and Jupyter Notebook on macOS

1. Download the GUI of the macOS installer for the respective version of Python
2. After downloading, open the file and click on the Continue button to start the process.
3. Click the check box on the Read Me screen and then on the License screen
4. In the Select Destination window, click on **Install for me only** and then select Continue as shown in Figure 1.



Install Anaconda3

Select a Destination

How do you want to install this software?

Install for all users of this computer

Install for me only

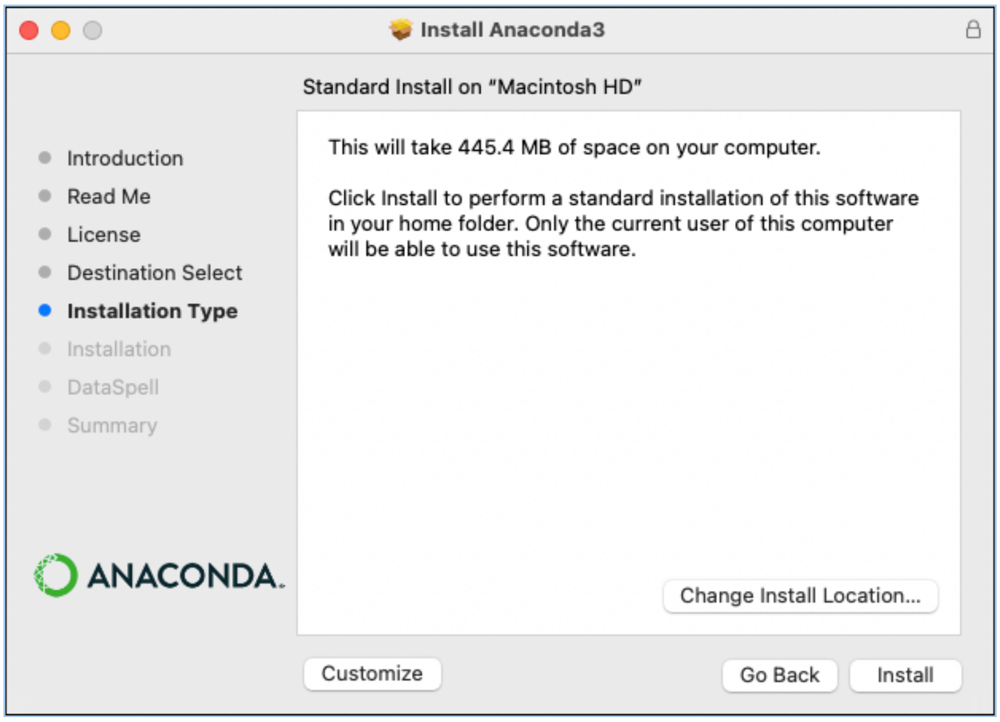
Install on a specific disk...

Installing this software requires 616.1 MB of space.

You have chosen to install this software in your home folder. Only the current user will be able to use this software.

Figure 1: Select Destination

5. In the Installation Type, Select the appropriate location for installation and click Install for Anaconda installation as shown in Figure 2.



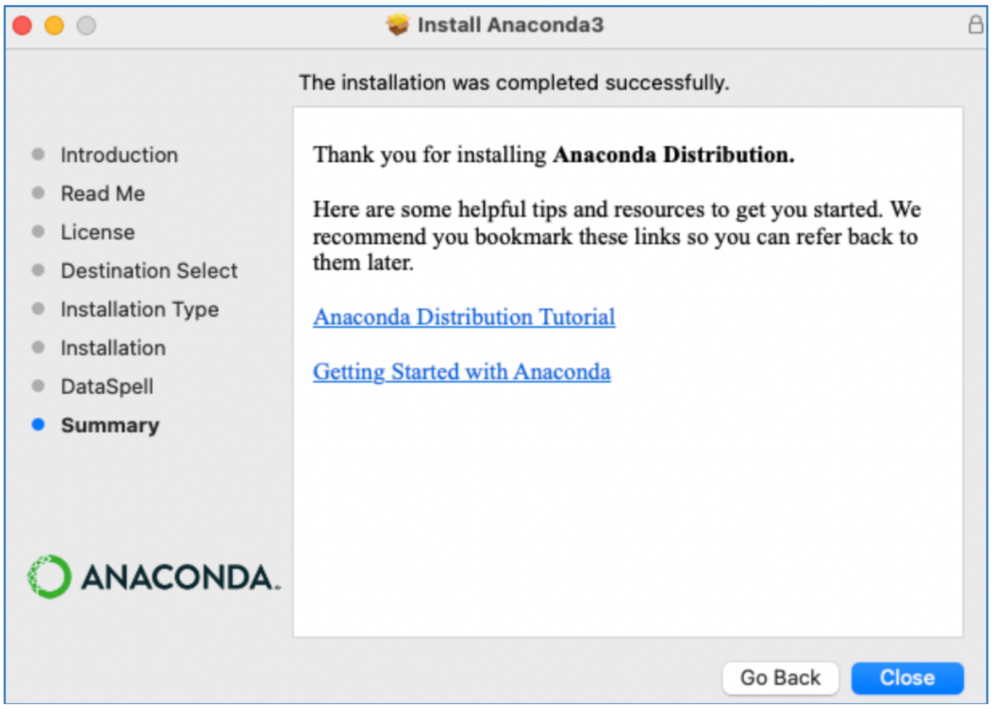
Standard Install on "Macintosh HD"

This will take 445.4 MB of space on your computer.

Click Install to perform a standard installation of this software in your home folder. Only the current user of this computer will be able to use this software.

Figure 2: Installation Type

6. After completion of installation, click Continue.
7. After completion of installation, click Continue, and after the successful installation the below summary screen will come up.



The installation was completed successfully.

Thank you for installing **Anaconda Distribution.**

Here are some helpful tips and resources to get you started. We recommend you bookmark these links so you can refer back to them later.

Anaconda Distribution Tutorial

Getting Started with Anaconda

Figure 3: Summary

3 Project Implementation Guide

All of the implementation, codes, packages, and logic that couldn't be presented in the report section are explained in this section.

3.1 Python Libraries

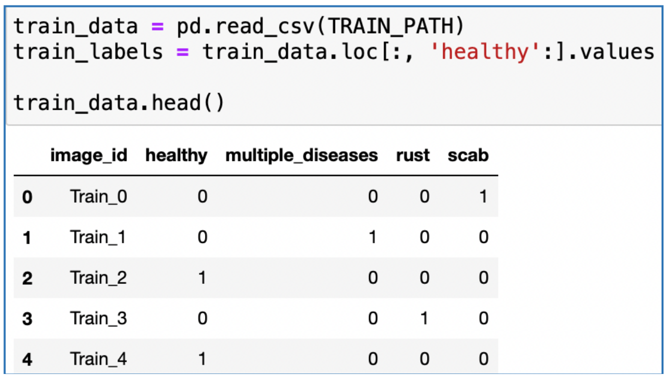
During the project, below are the important python libraries that are used to achieve the goal of identifying the apple foliar disease.

Table 3: Important Python Libraries

Libraries	Version
tensorflow	2.8.0
Pandas	1.4.1
sklearn	1.0.2
cv2	4.6.0
numpy	1.12.2
matplotlib	3.5.1
plotly	5.9.0
seaborn	0.11.2

3.2 Data understanding and Pre-processing

Understanding the data is necessary before applying a model to any dataset. For instance, it is important to know what kind of data we have and how well it is functioning.

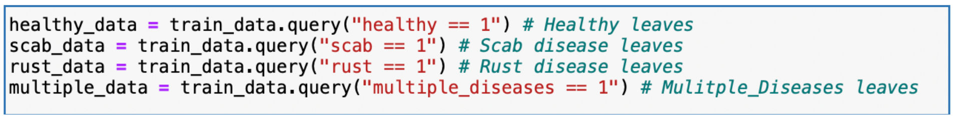


```
train_data = pd.read_csv(TRAIN_PATH)
train_labels = train_data.loc[:, 'healthy':].values

train_data.head()
```

```
   image_id  healthy  multiple_diseases  rust  scab
0  Train_0   0                        0     0     1
1  Train_1   0                        1     0     0
2  Train_2   1                        0     0     0
3  Train_3   0                        0     1     0
4  Train_4   1                        0     0     0
```

Figure 4: Label Extraction

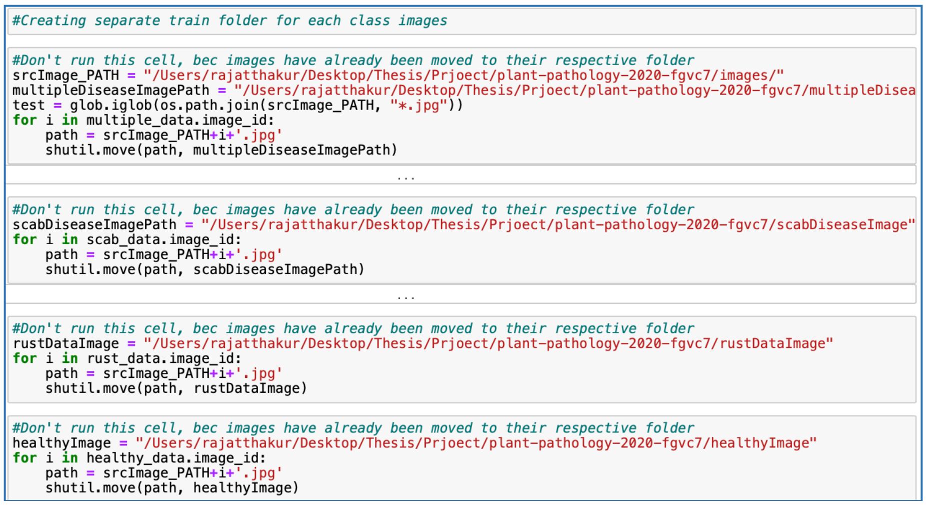
The below code is used to extract all the classes in respective class variable name.



```
healthy_data = train_data.query("healthy == 1") # Healthy leaves
scab_data = train_data.query("scab == 1") # Scab disease leaves
rust_data = train_data.query("rust == 1") # Rust disease leaves
multiple_data = train_data.query("multiple_diseases == 1") # Mulitple_Diseases leaves
```

Figure 5: Class variables

The image below demonstrates how distinct test and train folders were made for each class image. Before moving the dataset images to other folders, the code below was only executed once. For implementing LittleVGG and MobileNet models, separate folders for all the images have been created programmatically so that it would be easy to recognise classes and diseases by the models.



```
#Creating separate train folder for each class images

#Don't run this cell, bec images have already been moved to their respective folder
srcImage_PATH = "/Users/rajatthakur/Desktop/Thesis/Prjoect/plant-pathology-2020-fgvc7/images/"
multipleDiseaseImagePath = "/Users/rajatthakur/Desktop/Thesis/Prjoect/plant-pathology-2020-fgvc7/multipleDisea
test = glob.iglob(os.path.join(srcImage_PATH, "*.jpg"))
for i in multiple_data.image_id:
    path = srcImage_PATH+i+'.jpg'
    shutil.move(path, multipleDiseaseImagePath)
                                        ...
#Don't run this cell, bec images have already been moved to their respective folder
scabDiseaseImagePath = "/Users/rajatthakur/Desktop/Thesis/Prjoect/plant-pathology-2020-fgvc7/scabDiseaseImage"
for i in scab_data.image_id:
    path = srcImage_PATH+i+'.jpg'
    shutil.move(path, scabDiseaseImagePath)
                                        ...
#Don't run this cell, bec images have already been moved to their respective folder
rustDataImage = "/Users/rajatthakur/Desktop/Thesis/Prjoect/plant-pathology-2020-fgvc7/rustDataImage"
for i in rust_data.image_id:
    path = srcImage_PATH+i+'.jpg'
    shutil.move(path, rustDataImage)

#Don't run this cell, bec images have already been moved to their respective folder
healthyImage = "/Users/rajatthakur/Desktop/Thesis/Prjoect/plant-pathology-2020-fgvc7/healthyImage"
for i in healthy_data.image_id:
    path = srcImage_PATH+i+'.jpg'
    shutil.move(path, healthyImage)
```

Figure 6: Moving Images to separate folder for LittleVGG and MobileNet

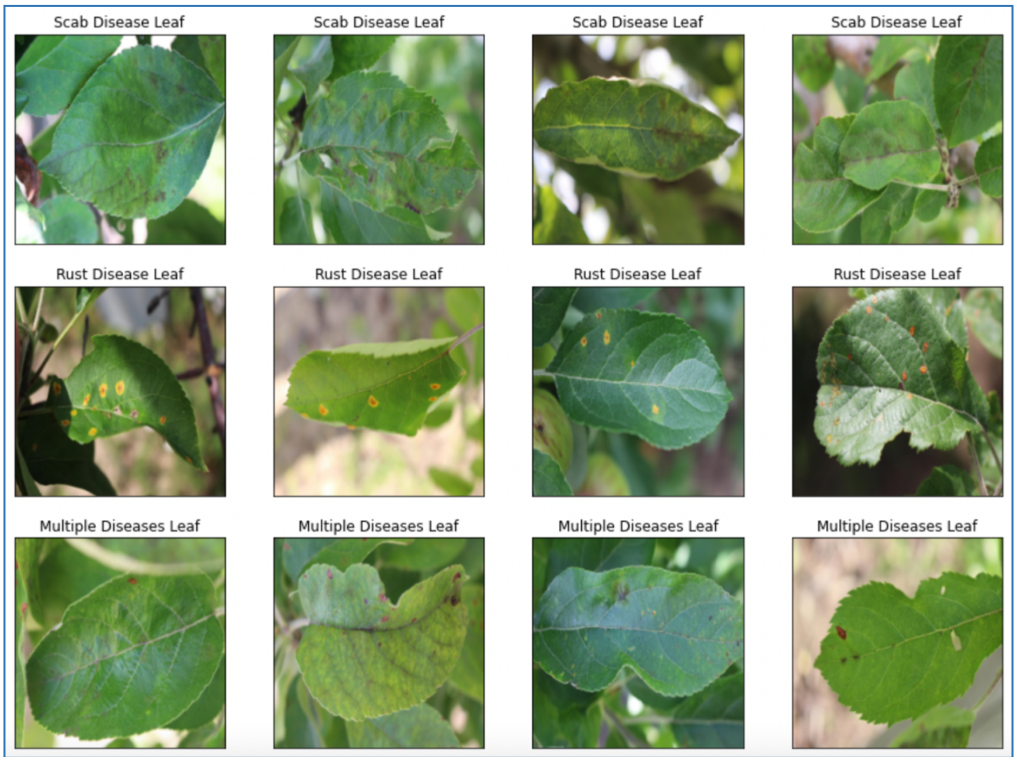


Figure 7: Sample Diseases images

By removing excess noise and smoothing the images, a Canny Edge Detector method has been utilized to identify the edges of an input image as shown below figure.

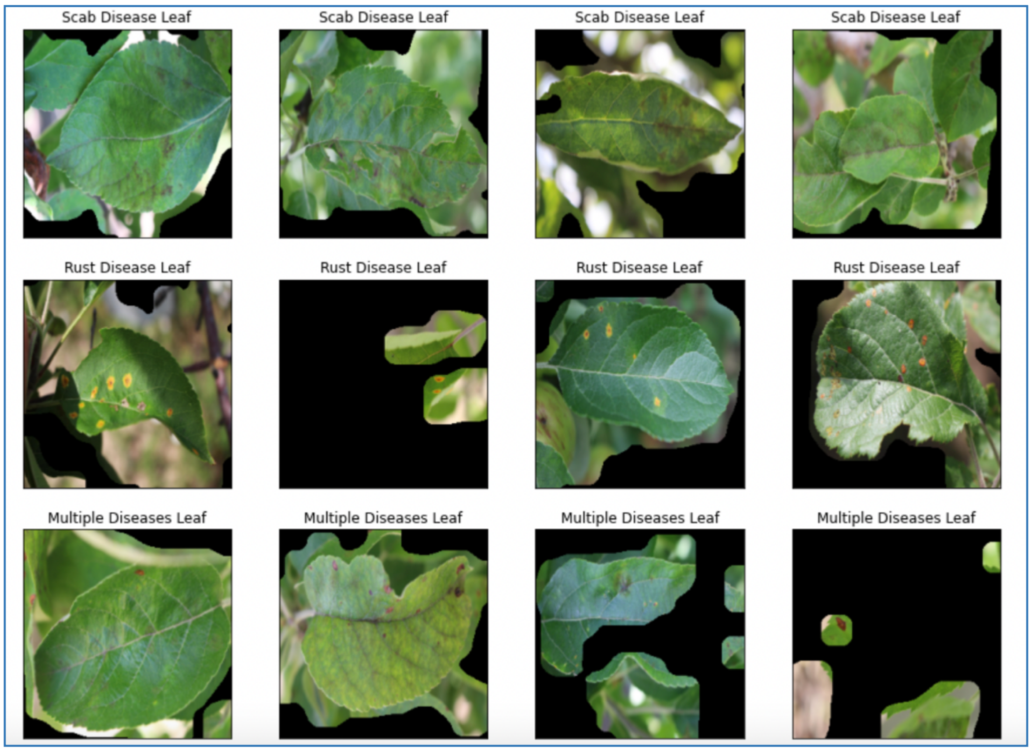


Figure 8: Canny Edge Detection Sample Images

GAN Model Implementation sample: The below screenshot shows the GAN model-generated sample image to resolve the data imbalance issue.

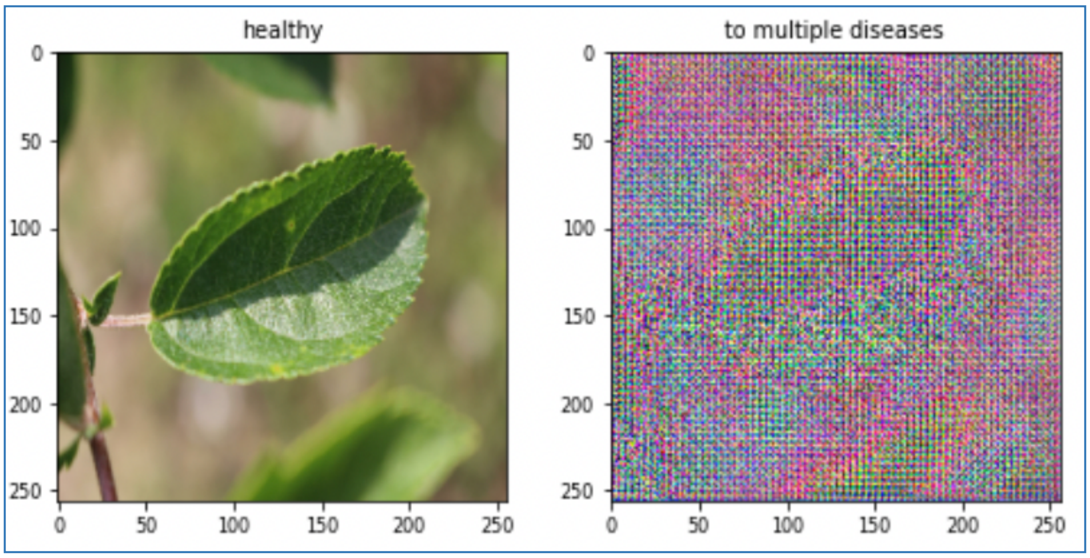
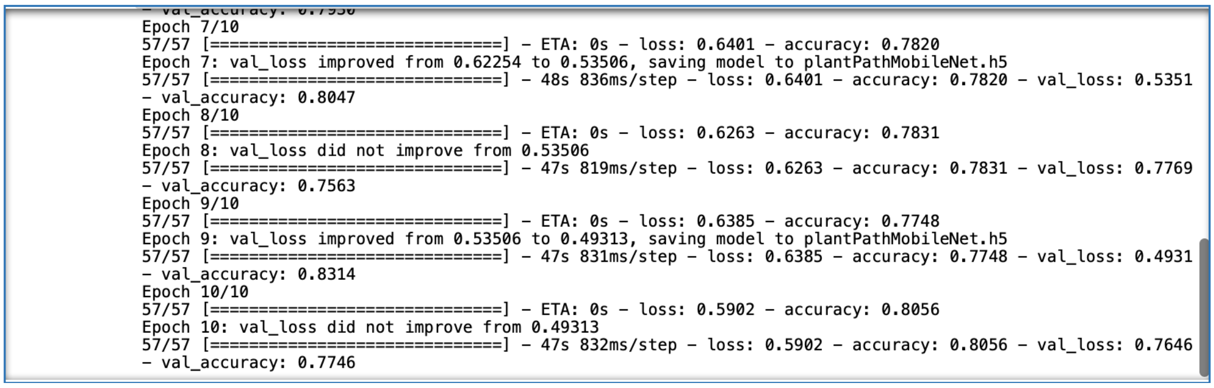


Figure 9: GAN, healthy to multiple diseases image

3.3 Model Implementations

Two of the top models are reviewed in this section and their epochs and degree of accuracy are displayed below.

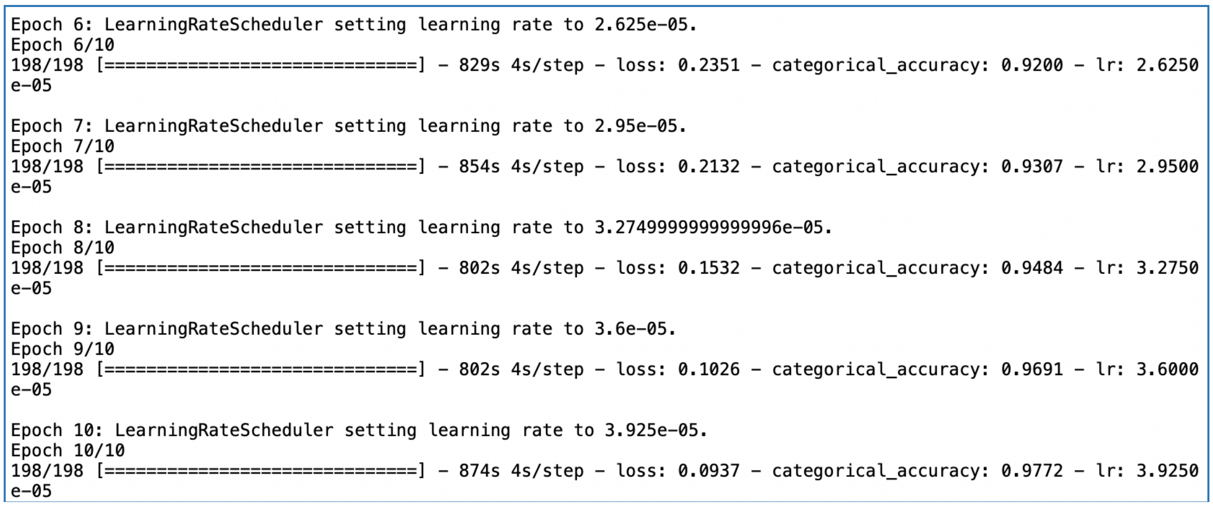
MobileNet Implementation Output: The second best model for identifying the foliar disease in apple trees is MobileNet Model, and below is the epoch screenshot attached for reference.



```
- val_accuracy: 0.7930
Epoch 7/10
57/57 [==============================] - ETA: 0s - loss: 0.6401 - accuracy: 0.7820
Epoch 7: val_loss improved from 0.62254 to 0.53506, saving model to plantPathMobileNet.h5
57/57 [==============================] - 48s 836ms/step - loss: 0.6401 - accuracy: 0.7820 - val_loss: 0.5351
- val_accuracy: 0.8047
Epoch 8/10
57/57 [==============================] - ETA: 0s - loss: 0.6263 - accuracy: 0.7831
Epoch 8: val_loss did not improve from 0.53506
57/57 [==============================] - 47s 819ms/step - loss: 0.6263 - accuracy: 0.7831 - val_loss: 0.7769
- val_accuracy: 0.7563
Epoch 9/10
57/57 [==============================] - ETA: 0s - loss: 0.6385 - accuracy: 0.7748
Epoch 9: val_loss improved from 0.53506 to 0.49313, saving model to plantPathMobileNet.h5
57/57 [==============================] - 47s 831ms/step - loss: 0.6385 - accuracy: 0.7748 - val_loss: 0.4931
- val_accuracy: 0.8314
Epoch 10/10
57/57 [==============================] - ETA: 0s - loss: 0.5902 - accuracy: 0.8056
Epoch 10: val_loss did not improve from 0.49313
57/57 [==============================] - 47s 832ms/step - loss: 0.5902 - accuracy: 0.8056 - val_loss: 0.7646
- val_accuracy: 0.7746
```

Figure 10: MobileNet Model output for 10 Epoch

EfficientNet Implementation Output: According to our research, EfficientNet performed better than LittleVGG and MobileNet, the other two models. Overall the EfficientNet model identified the apple foliar diseases with a 97.72% accuracy rate.

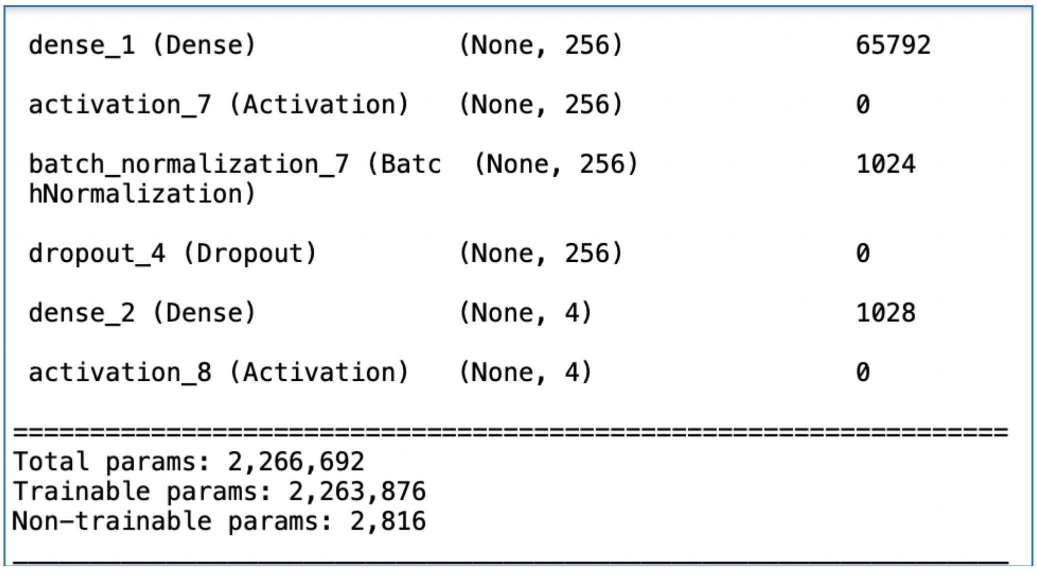


```
Epoch 6: LearningRateScheduler setting learning rate to 2.625e-05.
Epoch 6/10
198/198 [==============================] - 829s 4s/step - loss: 0.2351 - categorical_accuracy: 0.9200 - lr: 2.6250
e-05

Epoch 7: LearningRateScheduler setting learning rate to 2.95e-05.
Epoch 7/10
198/198 [==============================] - 854s 4s/step - loss: 0.2132 - categorical_accuracy: 0.9307 - lr: 2.9500
e-05

Epoch 8: LearningRateScheduler setting learning rate to 3.2749999999999996e-05.
Epoch 8/10
198/198 [==============================] - 802s 4s/step - loss: 0.1532 - categorical_accuracy: 0.9484 - lr: 3.2750
e-05

Epoch 9: LearningRateScheduler setting learning rate to 3.6e-05.
Epoch 9/10
198/198 [==============================] - 802s 4s/step - loss: 0.1026 - categorical_accuracy: 0.9691 - lr: 3.6000
e-05

Epoch 10: LearningRateScheduler setting learning rate to 3.925e-05.
Epoch 10/10
198/198 [==============================] - 874s 4s/step - loss: 0.0937 - categorical_accuracy: 0.9772 - lr: 3.9250
e-05
```

Figure 11: EfficientNet output for 10 Epoch

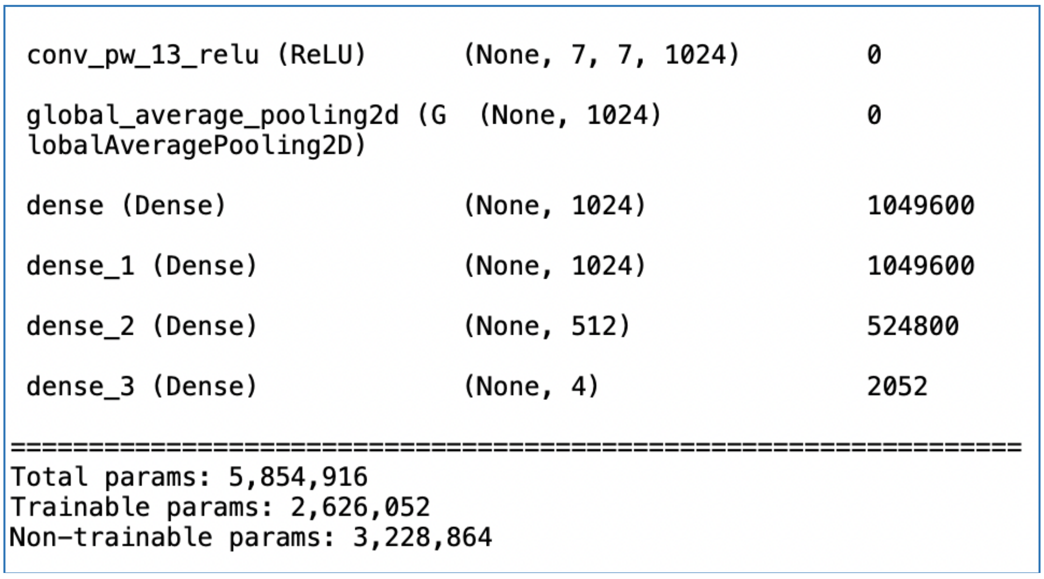
3.4 Trainable Parameters

Below is the summary of the developed LittleVGG and MobileNet models.



```
 dense_1 (Dense)             (None, 256)               65792

 activation_7 (Activation)   (None, 256)               0

 batch_normalization_7 (Batc  (None, 256)              1024
 hNormalization)

 dropout_4 (Dropout)         (None, 256)               0

 dense_2 (Dense)             (None, 4)                 1028

 activation_8 (Activation)   (None, 4)                 0

=================================================================
Total params: 2,266,692
Trainable params: 2,263,876
Non-trainable params: 2,816
_________________________________________________________________
```

Figure 12: LittleVGG Trainable params



```
 conv_pw_13_relu (ReLU)      (None, 7, 7, 1024)        0

 global_average_pooling2d (G  (None, 1024)             0
 lobalAveragePooling2D)

 dense (Dense)               (None, 1024)              1049600

 dense_1 (Dense)             (None, 1024)              1049600

 dense_2 (Dense)             (None, 512)               524800

 dense_3 (Dense)             (None, 4)                 2052

=================================================================
Total params: 5,854,916
Trainable params: 2,626,052
Non-trainable params: 3,228,864
```

Figure 13: MobileNet Trainable params

3.5 Model Implementation Prediction

Here are a few examples of sample images of apple foliar diseases that the models had predicted.

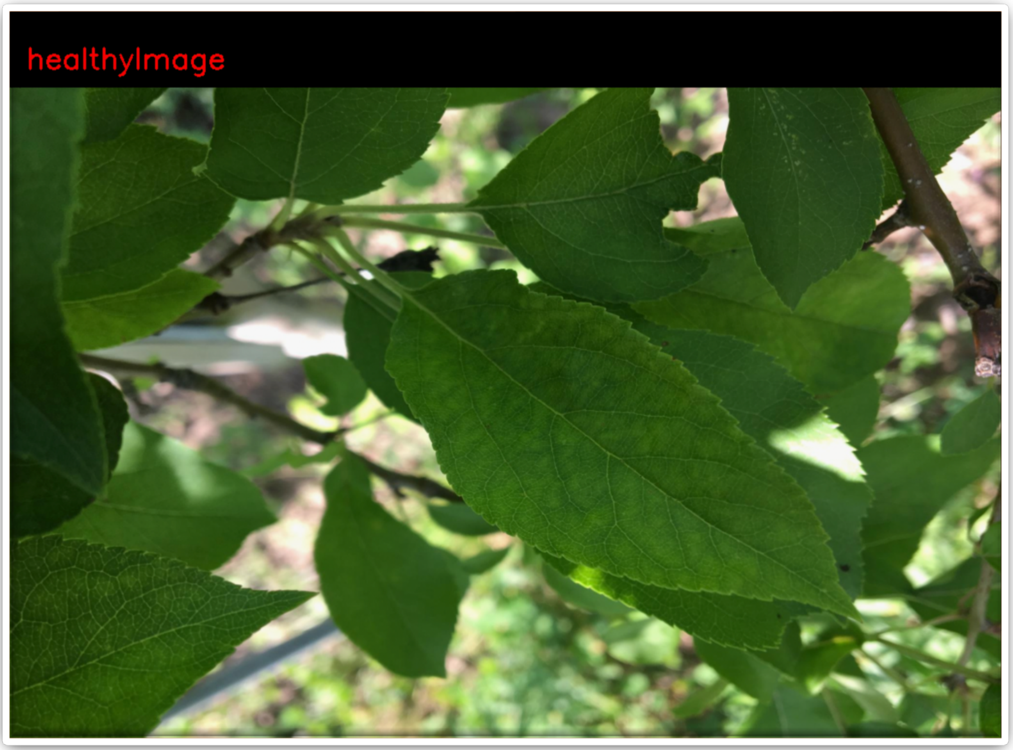


Figure 14: Prediction of Healthy leaf image

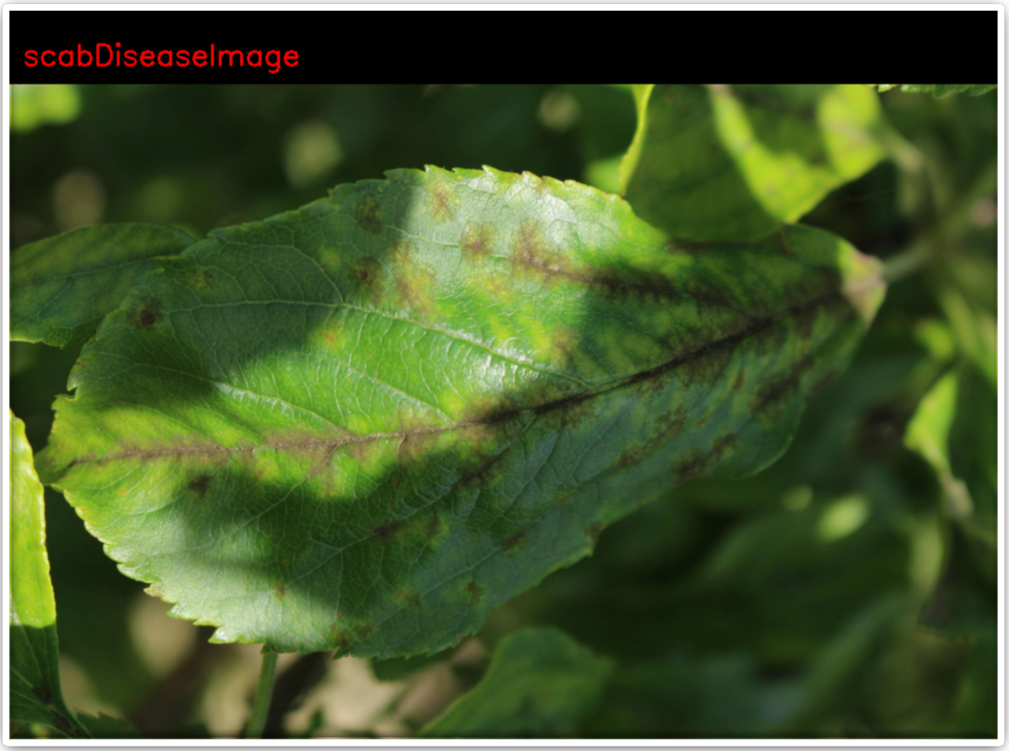


Figure 15: Prediction of Scab leaf image

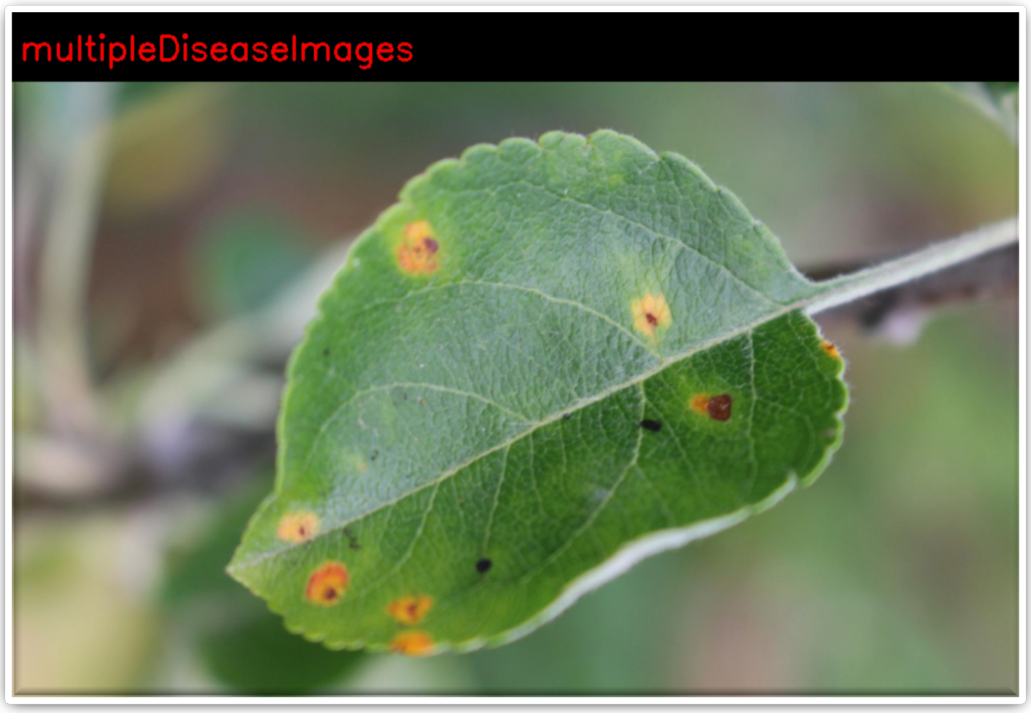


Figure 16: Prediction of Multiple Disease leaf image

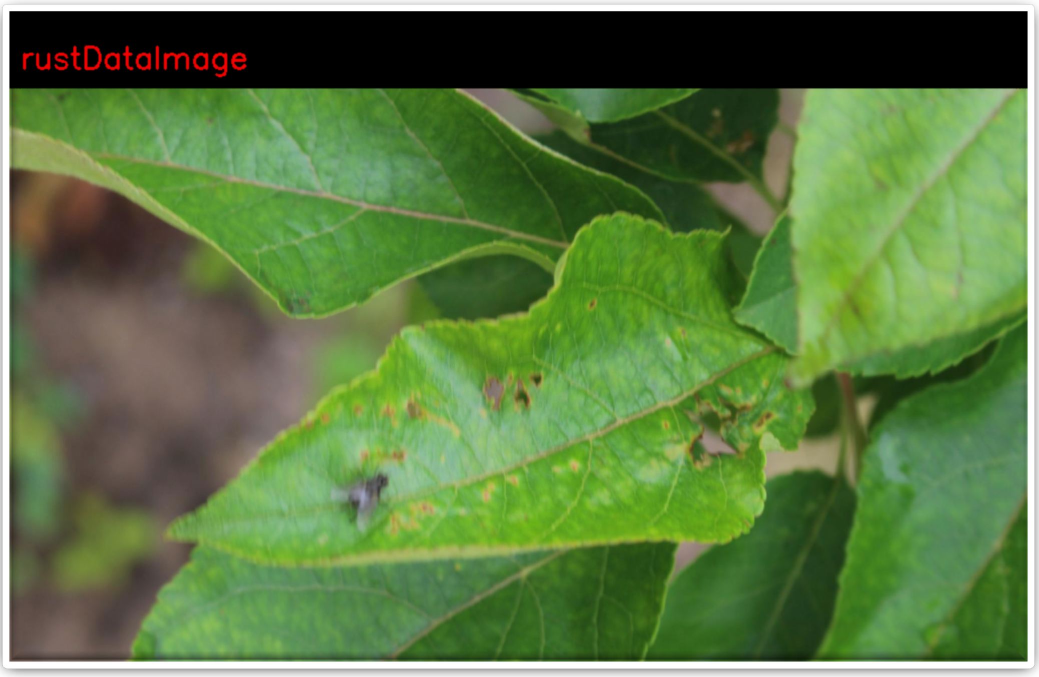


Figure 17: Prediction of Rust Disease leaf image

4 Comparison of Developed Models

Below is the table which shows the accuracy of the developed model used for this research.

Table 4: Model comparision

Model Name	**Accuracy**
LittleVGG	41.39%
MobileNet	80.56%
EfficientNet	97.72%

5 Conclusion

For the detection of foliar disease in apple trees, transfer learning algorithms including LittleVGG, MobileNet, and EffiecientNet have been deployed. It has been debated which approach may help with the identification of foliar disease more precisely.

6 References

Thapa, Ranjita et al. "The Plant Pathology Challenge 2020 data set to classify foliar disease of apples." Applications in plant sciences vol. 8,9 e11390. 28 Sep. 2020, doi:10.1002/aps3.11390